Задание 1.

К каждой таблице в отдельности написать запрос, возвращающий сумму продаж на каждую дату.

Таб. 1

|  |  |
| --- | --- |
| **tbl\_sales** | |
| **date (PK)** | **amount** |
| 01.01.2016 | 100 |
| 02.01.2016 | 150 |
| 03.01.2016 | 50 |
| 04.01.2016 | 100 |
| 05.01.2016 | 200 |
| 06.01.2016 | 95 |

Таб. 2

|  |  |
| --- | --- |
| **tbl\_sales** | |
| **date** | **amount** |
| 01.01.2016 | 100 |
| 02.01.2016 | 150 |
| 03.01.2016 | 50 |
| 03.01.2016 | 100 |
| 05.01.2016 | 200 |
| 05.01.2016 | 95 |

select \*  
from tbl\_sales  
  
Комментарий к первому запросу: у нас есть сущность с именем tbl\_sales. Сущность содержит n строк. date – это естественный первичный ключ, который уникально определяет строку в таблице, таким образом, группировка не требуется, так как каждая дата уникальна и запрос будет являться простым select’ом для получение суммы накопленным итогом.

select date, sum(amount) as sales  
from tbl\_sales  
group by date   
  
Комментарий к второму запросу: здесь date уже не является естественным первичным ключом, а значит требуется группировка по каждой дате, чтобы получить сумму продаж накопленным итогом за каждую дату. Если других полей в таблице нет, то можно предположить, что есть суррогатный ключ для этой таблицы или другой естественный ключ. Результат можно видеть снизу.

|  |  |
| --- | --- |
| **tbl\_sales** | |
| **date** | **amount** |
| 01.01.2016 | 100 |
| 02.01.2016 | 150 |
| 03.01.2016 | 150 |
| 05.01.2016 | 295 |

Задание 2.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Пользователи | |  | Заказы | | |
| **tbl\_member** | |  | **tbl\_sales** | | |
| **member\_id** | **gender\_id** |  | **order\_id** | **member\_id** | **amount** |

a) Написать один запрос, возвращающий кол-во пользователей, покупателей, заказов и сумму всех заказов.

Несколько запросов с объединением:   
--Количество пользователей--  
select count(member\_id) as result   
from tbl\_member  
union   
--Количество покупателей--  
select count(distinct member\_id)   
from tbl\_sales  
union   
--Количество заказов--  
select sum(count\_order\_member)   
from (select member\_id, count(order\_id) as count\_order\_member  
from tbl\_sales  
group by 1)  
--Сумма всех заказов--   
select sum(amount)   
from tbl\_sales  
  
Результирующая таблица должна получится такая:

|  |
| --- |
| result |
| Кол-во пользователей |
| Кол-во покупателей |
| Кол-во заказов |
| Сумма всех заказов |

Комментарий к запросу: общее количество пользователей можно посчитать просто посчитав все строки в таблице tbl\_member – это будут пользователи часть которых сделала заказ, а часть которых заказ не сделала. Количество уникальных покупателей можно посчитать в таблице tbl\_sales, где пользователь – это человек, который сделал заказ. Количество тоже можно посчитать через группировку по каждому клиенту. Сумма всех заказов также считается аналогично.

Один запрос:   
select count(member\_id), count(order\_id), sum(amount)  
from tbl\_sales   
left join tbl\_member on tbl\_member.member\_id = tbl\_sales.member\_id

Комментарий к запросу: так как покупатель, как описал выше, человек, который сделал заказ, то количество заказов = количеству покупателей   
  
b) Написать запрос, возвращающих всех пользователей, оформивших более трех заказов на общую сумму не менее 10000 рублей.

select member\_id, sum(amount), count(order\_id)  
from tbl\_member  
inner join tbl\_sales on tbl\_sales.member\_id = tbl\_member.member\_id   
group by member\_id  
having sum(amount) > 10000 and count(order\_id) > 3

c) Написать запрос, возвращающий на каждого пользователя по три самых дорогих заказа.

with client\_order as (  
select member\_id, order\_id, sum(amount)  
from tbl\_sales  
group by 1, 2  
order by sum(amount) asc   
)  
  
select   
 member\_id,   
 order\_id,   
 row\_number() over (partition by member\_id) as rn   
from client\_order  
where rn <= 3

Задание 3.

a) Есть ли ошибки в следующих запросах:

1.

select id

,YEAR(BillingDate) as BillingYear   
  
аналогичен будет запрос datepart(year, BillingDate::date) as BillingYear  
вернет значение типа year от времени, поэтому ошибок нет   
  
или extract(year from BillingDate::date)

from invoices

where BillingYear >= 2010

Ошибок нет, аллиясы можно фильтровать

2.

select id

,YEAR(BillingDate) as BillingYear

from invoices

order by BillingYear desc аналогично и здесь нет ошибок, так как время можно сортировать

b) Для чего используется оператор UNION? В чем отличие UNION от UNION ALL?

UNION – это оператор объединения запросов (он по умолчанию удаляет дубликаты из объединения)   
UNION ALL – это оператор, который сохраняет дубликаты при объединении

c) Перечислите все виды соединений, которые вам известны, и расскажите об их отличиях.

INNER JOIN – внутреннее соединение по ключам  
LEFT JOIN – соединяет значения из левой таблицы со всеми совпадающими справа, если нет сопоставления справа, то NULL, но значение слева сохраняется  
RIGHT JOIN – аналог left join, если поменять таблицы местами   
CROSS JOIN – декартово произведение   
FULL JOIN – left join + right join   
SELF JOIN – соединение на саму себя (нужно для статистических исследований)

Задание 4.

|  |  |  |
| --- | --- | --- |
| **id** | **name** | **referred\_by** |
| 1 | John | null |
| 2 | Jane | null |
| 3 | Anne | 2 |
| 4 | Eric | null |
| 5 | Rat | 1 |
| 6 | Alice | 2 |

select name

from customers

where referred\_by <> 2

Какой результат вернет этот запрос?

Так как null – это ничто (условно), то мы не можем сравнивать что-то с ничто, поэтому null не учитываются при фильтровке   
  
Ответ: Rat